# RANGE AND MISSION SCHEDULING AUTOMATION USING COMBINED AI AND OPERATIONS RESEARCH TECHNIQUES

Mansur Arbabi, Ph.D.
Michael Pfeifer
International Business Machines Corporation
Federal Systems Division
18100 Frederick Pike
Gaithersburg, Maryland 20879

## ABSTRACT

Ground-based systems for Satellite Command, Control, and Communications (C ) operations require a method for planning, scheduling and assigning the range resources such as: antenna systems scattered around the world, communications systems, and personnel. The method must accommodate user priorities, last minute changes, maintenance requirements, and exceptions from nominal requirements.

Described are computer programs which solve 24-hour scheduling problems, using heuristic algorithms and a real-time interactive scheduling process. The computer utilized is an IBM System/370, Model 3081, and an IBM 3279 color graphic display.

## INTRODUCTION

Ground-based systems for Satellite Command, Control and Communications (C ) operations require a method for planning, scheduling and assigning the range resources such as: antenna systems scattered around the world, communications systems, and personnel. The method must accommodate user priorities, last minute changes, maintenance requirements, and exceptions from nominal requirements.

Recognizing this need and its potential application to programs such as Data System Modernization (DSM) for the U.S. Air Force Satellite Control Network, IBM has pursued an Independent Research and Development (IRAD) effort to investigate a means of automating the scheduling of range resources for a satellite ground-based C system.

In existing systems, schedules typically are manually prepared for times in the future ranging from many months to one day, and, in some cases, near real-time changes must be accommodated. This manual scheduling is a very labor-intensive process and, at best, it offers scheduling accuracy of one minute. Over the past few years, the number and complexity of satellites have increased significantly. These increases have strained the capacity of manual scheduling, necessitating the analysis of automated scheduling techniques.

This article addresses the results of the three-year research project undertaken by IBM's Federal System Division at Gaithersburg, Maryland. Described is a computer program which solves 24-hour scheduling problems, using heuristic algorithms, in less than two minutes on an IBM System/370 Model 3081 using an APL interpreter under MVS. This program provides results in user selectable time unit granularity, and with accuracy constrained by computer precision limits.

## RANGE SCHEDULING

The range scheduling problem involves allocation of range resources to satellite operations. The allocation process is done for planned activities which range from six months in the future to near real time. The problem is complicated by time constraints and last minute modifications. The range scheduling function will be subjected to increasing pressure as the number of space vehicles increases. The severity of this problem is increased by other factors such as the addition of antennas and other resources at existing sites, reduced turnaround time, and increased demand for shared resources. It is also important to take full advantage of future increases in computational capability to sustain a high level of system utilization.

There are several technical issues related to this effort. Many schedules are required to cover time periods from 24 hours to six months. The schedules satisfy different purposes and must be presented in appropriate levels of detail. User requests for services are not static, and provisions must be made for changes on short notice. Many priorities must be accommodated. Allowances for schedule modifications due to malfunctions in either the satellite or the ground support equipment must be taken into account. A method of presenting automatically developed schedules in a meaningful way is essential to the success of automated range scheduling. It is expected that the users will interact with the system to generate and modify schedules.

As a background for this task, a thorough investigation of previous work on scheduling of ground resources in support of satellite operations was made. It included both NASA and DoD scheduling efforts.

The only automated scheduling found was for small problems (less than 50 requests, two antennas with short windows). It was also found that many agencies within DoD and NASA are interested in a solution to the same problem and are investigating generalized scheduling techniques.

Objective

The objective of this research was to determine the feasibility of computer-generated range scheduling and to demonstrate such a capability.

Approach

Many of the functions performed by the schedulers can be performed readily by computers. Certain other functions require further research to bring them closer to automation. The functions of the scheduling process considered for this study are those which had not been previously automated. These are:

-request processing

-production of weekly schedules

-scheduling conflict identification and resolution

-production of daily support schedules

-real time schedule changes

An overview of the scheduling function is shown in Figure 1.

Remote telemetry, tracking and command antennas, located around the world, are used to communicate with satellites when they are within line-of-sight range. Each antenna can "contact", at most, one satellite at a time.

Before an antenna can be used for a contact, there is a certain amount of "set-up" time or "turnaround" time that is required by the ground crew to reconfigure the antenna system.

Users place demands on the system by requesting that blocks of (contiguous) antenna time (also known as contact times) be allocated to their satellites. These requests can take various forms. Often, but not always, the specific antenna and exact time for the contact are not specified. An antenna preference, a contact priority, and an earliest and latest time for the contact may, however, be given.

Users may request periodic contacts or multiple simultaneous contacts. Finally, users may request continuous contact with their satellite over long periods of time. These requests can be met by piecing together overlapping contacts from multiple ground antennas. The process is called a "hot handoff".

There are also non-satellite support requests which are confined to a single antenna for such purposes as preventive maintenance.

Since normally more ground antenna support time is requested by the various users than is available, conflicts in the user requested support must be resolved. A good scheduling algorithm can minimize these conflicts and help alleviate this situation.

When stated generally, the scheduling problem is quite difficult. Linear programming techniques have little (practical) pay-off for the scheduler. The basic problem can be reduced to a mixed integer linear program, but

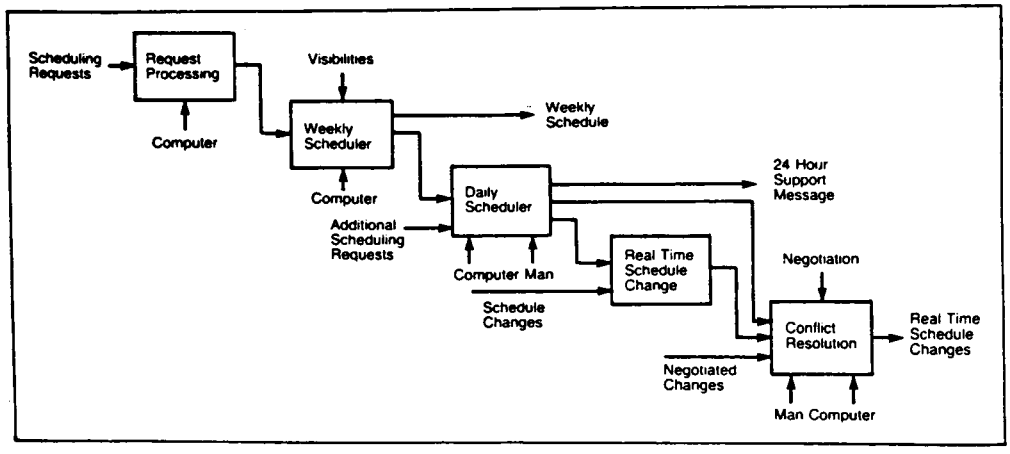



Figure 1. Overview of Scheduling Functions

this is of little (practical) consequence due to the large number of variables that are required. A heuristic technique would appear to be the only feasible approach.

Fortunately, real-world problems have additional attributes. Satellites tend to fall into one of three (almost) disjoint classes. Each class has it own special contact request pattern.

Low altitude satellites have an apogee of under 500 miles. They are visible over a ground antenna for only about ten minutes before they disappear over the horizon. Contacts, when requested, are for the entire time that the satellite is within the line of sight of an antenna. Most of the high priority requests come from this class.

Medium altitude satellites have an altitude which averages 12,000 miles and maintain line of sight with an antenna for up to 11 hours. Contacts re generally requested for 10 minutes' duration within a 45-minute window during which users may prefer a particular antenna to make the contact.

Near synchronous satellites have altitudes in the vicinity of 22,000 miles. If they are at the right position on earth, antennas can maintain line-of-sight contact for many hours (or continuous in the case of truly synchronous satellites). Users request varying length contact times and "hot handoffs" generally come from this class of satellites.

Non-satellite support requests are station-specific, but generally are fairly flexible as to when they are scheduled. The length of the support period ranges from 10 minutes to several hours.

The most significant accomplishment of this effort was the development of a new continuous time scheduling (CTS) algorithm for range scheduling. As described below, it is believed that the CTS algorithm demonstrates, for the first time, the feasibility of providing effective automation support to the complex scheduling operation.

## BACKGROUND OF THE SCHEDULING PROBLEM

The CTS algorithm is the result of earlier work on scheduling that began in 1981. Initially, an in-depth review of the manual scheduling techniques was conducted. Not surprisingly, they were found to be very sophisticated. The scheduling problems encountered were very complex. Typically, they involved as many as 300 requests to be satisfied by as many as 14 antennas during a 24-hour period. The numbers are increasing year by year. Working over many years, manual range scheduling personnel have developed powerful tools for handling these requirements and have evolved a complex set of priorities, rules, and exceptions. Most of these have not been formally documented, but are learned by extensive on-the-job training.

On the average, the operationally certified range scheduling personnel each have more than ten years of experience. By observing current procedures over several weeks, including several continuous 24-hour periods, an appreciation was gained for the problem and for the sophistication and limitations of manual scheduling methods.

In parallel, an extensive survey of existing automated scheduling systems was conducted. Reviewing some commercial, DoD and NASA systems, it was found that none was suitable for the scheduling loads and complexity needed.

Accordingly, IBM's efforts were directed to develop a new approach. Initially, so-called mathematical programming models were considered that attempted to establish optimum schedules by simultaneously allocating resources to all the space vehicles. It was determined that such models were feasible for scheduling fewer than 50 requests, but that the storage requirements and run times associated with larger numbers were unacceptably large, increasing exponentially with the number of requests.

Next, several heuristic models that attempted to "duplicate" the scheduling rules used by the manual schedules were developed. After investigating these approaches, a so-called "discrete laxity algorithm" was devised. By scheduling one satellite vehicle at a time, this approach could develop reasonably good schedules with long, but marginally acceptable, levels of computer resources (for example, storage and run times).

The results were documented to allow a comparison to the present manual process. It was learned that the principal limitations of the discrete laxity approach were the five-minute time unit granularity and the inability to handle special case requests.

The CTS algorithm was developed in 1983 to remedy the discrete laxity limitations. Figure 2 shows the paths by which the several heuristic and optimization techniques were combined to arrive at the CTS solution that uses the best features of both heuristic and optimization methods. Figure 3 summarizes the mixed integer equations utilized in the schedule optimization problem.

## A SIMPLE SCHEDULING PROBLEM

A simple example of the scheduling problem is presented. It consists of five requests: R1, R2, R3, R4 and R5. These requests are shown on the top diagram of Figure 4. Each request is specified by a duration, and a window having a start time and an end time. For example, request R1 has a duration of four time units and a window starting at time zero and ending at time 13. Request R5 has two separate windows. This example is a simple one since it is limited to a single antenna and to very simple request forms.

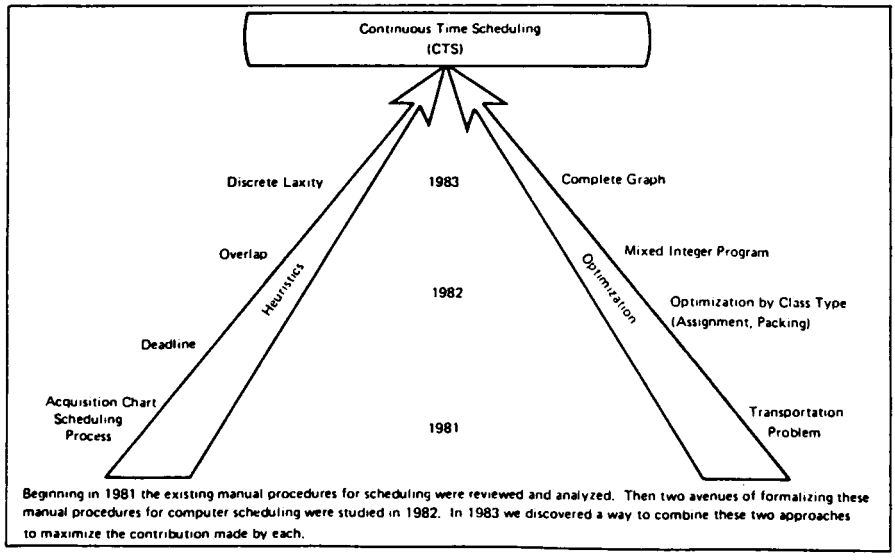



Figure 2. Progression of Scheduling Approaches

**Objective Function:**
Schedule as many requests as possible

$$\text{Maximize} \sum_{\forall K} V_K X_K$$

**Constraints:**
Schedule each request only once

$$\sum_{K \epsilon R_I} X_K \leq 1; \ \forall_I$$

Schedule each request within its window

$$A_K + S_K + C_K \leq B_K; \forall_K$$

Schedule requests to avoid concurrent resource use

$$A_J + S_J + C_J \leq A_K + S_K - T_K + M\delta_{JK} + M(2_i - X_J - X_K)$$

$$A_K + S_K + C_K \leq A_J + S_J - T_J + M(1-\delta_{JK}) + M(2 - X_J - X_K)$$

$$S_K \geq 0; \forall_K$$

$$X_K = (0, 1) \forall_K$$

$$\delta_{JK} = (0, 1) \forall (J,K) \epsilon P$$

**Definitions**

| | | |
|---|---|---|
| I | = | Request Index |
| J | = | Segment Index |
| K | = | Segment Index |
| $A_K$ | = | Beginning of Segment K |
| $B_K$ | = | End of Segment K |
| $C_K$ | = | Length of request on Segment K |
| $\delta_{JK}$ | = | 1 if request on K is started before request on J; 0 if otherwise |
| M | = | A large number (i.e., at least 3 times the scheduling period length) |
| P | = | The set of pairwise combinations of overlapping segment of each antenna |
| $R_I$ | = | Set of segments which service request I |
| $S_K$ | = | Offset between the beginning of Segment K and the beginning of its request |
| $T_K$ | = | Turn around time of the antenna on Segment K |
| $V_K$ | = | Preference value for scheduling a request on Segment K |
| $X_K$ | = | 1 if request I is scheduled on Segment K; 0 if otherwise |
| $\forall$ | = | for all |

This scheduling approach provides the best solution but unfortunately its computer running time and computer memory requirement grow exponentially with the number of input requests.

Figure 3. Mixed Integer Programming for Schedule Optimization

Before looking at the solution, it would be instructive for the reader to try various approaches such as "first come, first served" and "earliest deadline".

The steps necessary to arrive at a heuristic solution to this problem are shown in Figure 4. The top diagram is a barchart representing each request window. The problem is then represented in a graph form with all of its mathematical constraints. This graph formulation is then solved by evaluating the interaction between each request and the remainder of the requests.

The graph of the diagram of Figure 4 consists of nodes and links. The start time constraints for each request are shown by inequalities within the nodes of the graph; for example, the request R1 start time interval is:

$$0 \leq S \leq 9.$$

This means R1 could start any time between 0 and 9 and still remain in its window. When a request has more than one window segment, then for each segment there will be one time interval; for example, request R5, which has two segments, is shown with two sets of irregularities. The interaction between a set is shown by constraints on the links between the nodes of the graph, for example R1 and R3 start times $S$ and $S$ are free of conflict whenever:

$$S - S \leq 4$$

The last diagram in Figure 4 shows the solution tree. The request R2 is checked against R5, R1, R3 and R4 first for non-preemption and then for maximum laxity. Then the start time is selected for R2. In this manner we proceed from one request to the next until all requests are checked. The start times arrived at by this process are:

$$S = 9,\ S = 17,\ S = 13,\ S = 5,\ S = 0$$

Figure 5 shows the solution to the problem. Note that all the requests have been scheduled, and request R4 is scheduled in the middle of its window.

RESULTS

Figure 6 shows the results obtained in applying the algorithm to two representative problems. These scheduling results demonstrated that automated scheduling is indeed a viable alternative to manual scheduling. The average elapsed time required to develop a manual 24-hour schedule is 36 hours. The average number of weekly labor hours required to provide a manual 24-hour schedule is 645. The CTS algorithm scheduled greater than 98 percent of the input requests in less than three minutes of CPU run time on the IBM 3081 K32. It is expected that a 100 percent operationally optimum solution can be obtained with a small amount of manual intervention by the schedulers.

In Figure 7, computer run time is shown as a function of the number of requests schedule for a 24-hour sample problem.

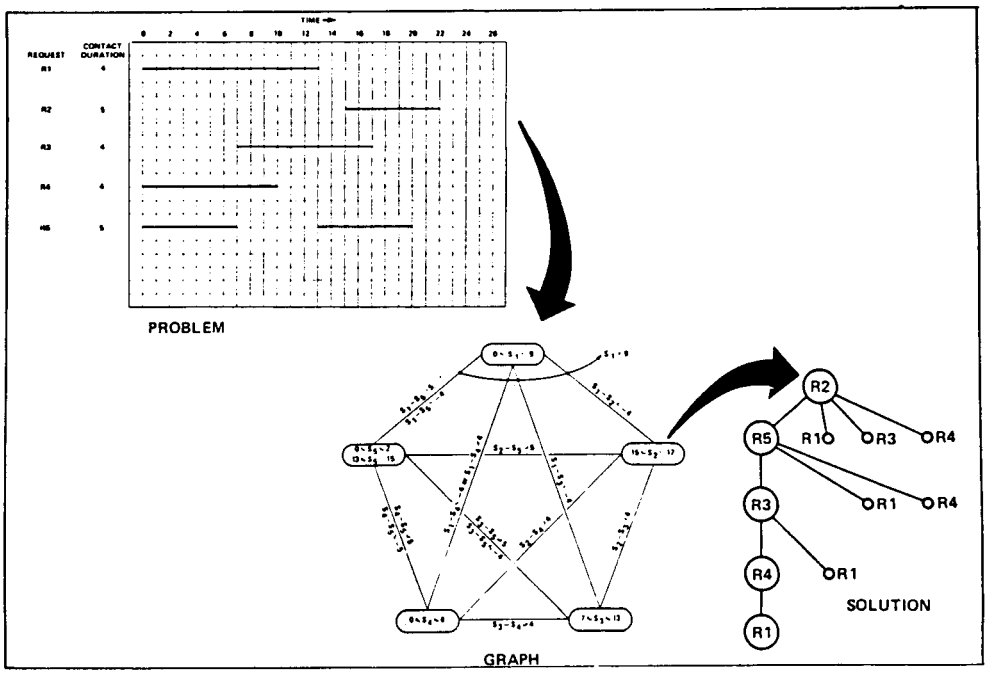



Figure 4. A Simple Scheduling Problem

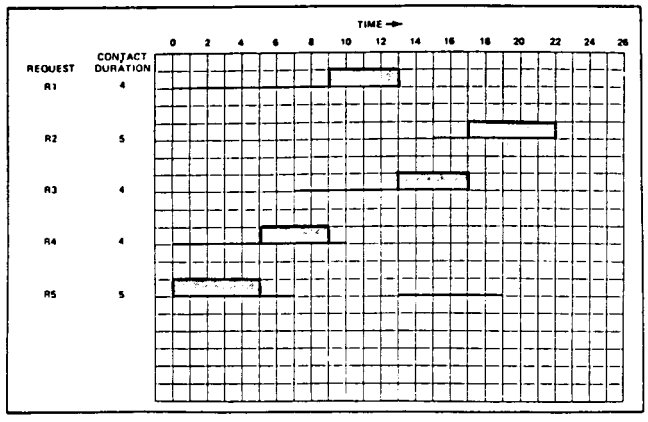
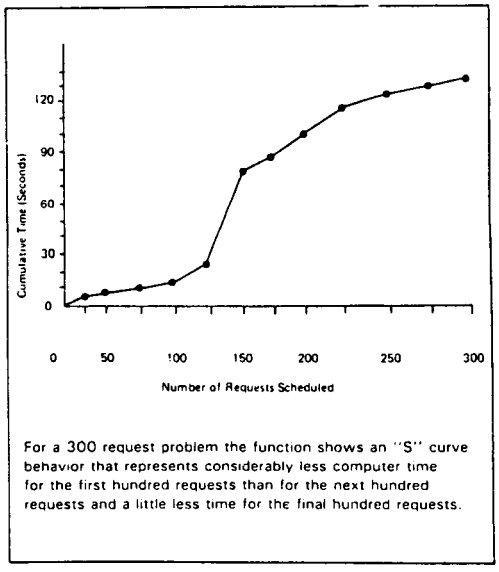



Figure 5. Solution--The Simple Scheduling Problem

| SAMPLE PROBLEMS | | |
|---|---|---|
| PROBLEM PARAMETERS (INPUT) | | |
| | Case 1 | Case 2 |
| • No. of Antennas | 12 | 12 |
| • No. of Visibilities | 617 | 933 |
| • Request Data: | | |
| – No. of Requests | 291 | 292 |
| – No. of Req. Segments | 1499 | 933 |
| – No. of Flight Req. | 240 | 245 |
| – No. of Non-Flight Req. | 51 | 47 |
| • Time Unit Granularity | 1 Min. | 1 Min. |
| OUTPUTS | | |
| • Number of Requests Scheduled | 286 | 289 |
| • Percent of Requests Scheduled | 98 | 99 |
| COMPUTER RUN TIME | | |
| • Processing Time Using MVS-APL on IBM 3081 K32 | 133 sec. | 62 sec. |
| In each case a 24-hour scheduling problem was solved in less than 3 minutes. | | |

Figure 6. CTS Algorithm Applied to Sample Problems




For a 300 request problem the function shows an "S" curve behavior that represents considerably less computer time for the first hundred requests than for the next hundred requests and a little less time for the final hundred requests.

Figure 7. Computer Running Time in Seconds vs. Requests Schedules (MVS-APL on IBM-3081

This algorithm has not yet been implemented in the operational Air Force Satellite Control Network System. It promises to be capable of handling scheduling problems with a high degree of efficiency and flexibility.

REFERENCES

1. IBM-FSD Data System Modernization: System Scheduling (AT13), Document No. C27-020-13-01, 16 April 1980.

2. IBM-FSD Data System Modernization, Generic Range Scheduling Functional Analysis, Contract F04690-81-C-003, 10 October 1982.

3. IBM-FSD Range Scheduling Automation, 2G59 IR&D Report, 1982.

4. IBM Mathematical Programming System Extended-Mixed Integer Programming Reference Manual, SH19-1099, August 1981.

5. IBM Mathematical Programming System Extended-Control Language SH19-1147, October 1978.